Lógica de programação

Profº Jânio Eduardo

janio.vasconcellos@gmail.com

Roteiro

- Objetivo geral
- Introdução Lógica de programação
  - Estrutura de comandos da Linguagem C
  - Saídas e entradas do console;
  - Função definição
  - Função printf();
  - Função scanf();
  - Execicios;
- Resumo da aula

Objetivo geral

Aprender a entender algoritmos para tomada de decisão

Professor Jânio Eduardo

Função conceito

- Funções em C são blocos de código que podem ser chamados a partir de qualquer lugar do programa, para executar uma determinada tarefa. Elas são essenciais em programas maiores, pois permitem a organização do código em blocos mais simples, isolando funcionalidades específicas em módulos independentes.

Algumas das principais vantagens de usar funções em C são

FBr
Faculdade Brasília

- **Modularização**: as funções permitem dividir o código em partes menores e mais organizadas, tornando-o mais fácil de entender, manter e modificar. Cada função pode ter uma tarefa específica e reutilizável, o que facilita a escrita de programas maiores e mais complexos.
- **Reutilização**: uma função pode ser chamada várias vezes a partir de diferentes partes do programa, o que evita a repetição desnecessária de código e torna o programa mais eficiente e legível.
- **Abstração**: uma função pode esconder os detalhes de implementação de uma determinada tarefa, fornecendo apenas uma interface mais simples e fácil de usar para o usuário. Isso permite que o usuário se concentre apenas na tarefa que precisa realizar, sem se preocupar com os detalhes de como ela é implementada.
- **Facilidade de teste**: funções bem definidas e isoladas podem ser facilmente testadas separadamente, o que ajuda a encontrar erros e a garantir a qualidade do código.

1.4. ESTRUTURA BÁSICA DE UM PROGRAMA C

```
Diretivas
Declarações de variáveis/constantes globais;
Prototipação de funções;
tipo_dado main(lista_parametros) /* Função Principal e Obrigatória */
{
        Declarações de variáveis/constantes locais;
        Comandos;
        Estruturas de controle (seleção e/ou repetição);
        Comentários;
        Chamada a outras funções;
}
tipo_dado func1(lista_parametros)
{
        Bloco de comandos;
}
tipo_dado func2(lista_parametros)
{
        Bloco de comandos;
}
•
•
•
tipo_dado funcN(lista_parametros)
{
        Bloco de comandos;
}
```

Função printf()

- Essa função tem por finalidade imprimir dados na tela. Isto é feito através da sintaxe:

printf(*"expressão de controle", lista de argumentos);*

- Na "expressão de controle" são inseridos todos os caracteres a serem exibidos na tela e/ou códigos de formatação, responsáveis por indicar o formato em que os argumentos devem ser impressos. Esses argumentos devem estar incluídos na "lista de argumentos" e caso contenha mais de um devem ser separados por virgula. A lista abaixo mostra os tipos de códigos de formatação permitido na linguagem C.

Tipos de códigos

%c	Caracteres simples
%d	Inteiros decimais com sinal (**decimal**)
%I	Inteiros decimais com sinal (**octal ou hexadecimal)**
%e	Notação cientifica (e minúsculo)
%E	Notacão cientifica (E maiúsculo)
%f	Ponto flutuante decimal
%g	Usa %e ou %f (qual for mais curto)
%G	Isa %E ou %F (qual for mais curto)
%o	Octal sem sinal
%s	Cadeia de caracteres
%u	Inteiros decimais sem sinal
%x	Hexadecimal sem sinal (letras minúsculas)
%X	Hexadecimal sem sinal (letras maiúsculas)
%%	Escreve o símbolo de porcentagem (%)

Simplificando

CÓDIGO	SIGNIFICADO
%c	Usado quando a função for exibir apenas um caractere (tipo *char*).
%f	Usado quando a função for exibir número com ponto flutuante (tipo *float*). Exemplo: 1.80
%s	Usado quando a função for exibir uma cadeia de caracteres, ou seja, uma ou várias palavras.
%d	Usado quando a função for exibir um número inteiro (tipo *int*).

Códigos especiais

Código especial	Descrição
\n	Nova linha
\t	Tab
\b	Retrocesso
\”	Aspas
\\	Barra
\f	Salta formulário
\0	Nulo

Função scanf()

- A função scanf() em C é usada para ler dados a partir da entrada padrão, que geralmente é o teclado. Ela permite que o programa receba dados digitados pelo usuário e armazene esses dados em variáveis específicas no programa.

scanf(formato, &variável);

Função scanf()

- A exemplo do *printf()*, o *scanf()* também utiliza os códigos de formatação.
- Enquanto no *printf()* esses códigos eram utilizados para indicar o formato dos dados a serem escritos, no *scanf()* esses mesmos códigos indicam o formato dos dados a serem lidos. O Quadro 5.4 exibe os códigos de formatação utilizados no *scanf()*. Note a semelhança com os códigos do *printf().*

Códigos de formato de leitura da função scanf()

Faculdade Brasília

CÓDIGO	FUNÇÃO
%c	Usado quando a função for armazenar um caractere (tipo char).
%f	Usado quando a função for armazenar um número com ponto flutuante, aquele valor com vírgula (tipo *float*).
%s	Usado quando a função for armazenar uma cadeia de caracteres, ou seja, uma ou várias palavras.
%d	Usado quando a função for armazenar um número inteiro (tipo int).

Comentários

- Quando desenvolvemos programas, devemos colocar textos que expliquem o raciocínio seguido durante seu desenvolvimento para que outras pessoas, ou nós mesmos, ao ler o programa mais tarde, não tenhamos dificuldades em entender sua lógica. Esses textos são chamados de comentários.
- Os comentários podem aparecer em qualquer lugar do programa. Em C, há dois tipos de comentários: os comentários de linha e os comentários de bloco.
- Os comentários de linha são identificados pelo uso de //. Assim, quando usamos // em uma linha, tudo o que estiver nessa linha depois do // são considerados comentários.

Exemplo de comentários

- Os comentários de bloco são iniciados por /* e finalizados por */. Tudo o que estiver entre esses dois símbolos são considerados comentários. Os comentários de bloco podem ocupar várias linhas.

```
int main( )
{
  int matricula= 2233;  /* podemos escrever cometários desta forma */
  float media_final=80.5; // ou apenas com duas barras no início do comentário
  char discipl[10]="Prog I";// a var discipl[10], pode armazenar até 10 caracteres
  printf ("O aluno matricula = %d \n", matricula);
  printf ("Disciplina = %s \n", discipl);
  printf ("Ficou com media = %f \n\n", media_final);
  system("pause");
  return(0);
}
```

Expressões aritiméticas

Operadores aritiméticos

Operador	Operação matemática
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
--	**Decremento Unário**
++	**Incremento Unário**
%	Resto da Divisão Inteira

Operador ++ e --

- O operador de incremento unário (++) incrementa de **1** o seu operando. Ou seja, se eu quiser aumentar em **1** o valor de uma variável **x**, posso fazer **x=x+1**;
- ou escrever simplesmente **x++**;. De forma análoga, o operador de decremento unário (--) decrementa de **1** o seu operando. Ou seja, se eu quiser diminuir de **1** o valor de uma variável **x**, posso fazer **x=x-1**; ou escrever simplesmente **x--**;.

Atividades, pode ser com várias funções ou vários programas

- Faça uma função cuja finalidade é calcular 8% de aumento sobre um salário.
- Faça uma função que leia um número inteiro e imprima seu antecessor e seu sucessor.
- Faça uma função que leia dois números reais e imprima a soma e a média aritmética desses números.
- Faça uma função que:
  - **a)** peça ao usuário para digitar um número inteiro;
  - **b)** armazene esse número numa variável chamada num1;
  - **c)** peça ao usuário para digitar outro número inteiro;
  - **d)** armazene esse número numa variável chamada num2;
  - **e)** some os valores e guarde o resultado numa variável chamada soma;
  - **f)** exiba os valores digitados;
  - **g)** exiba o resultado da soma
- Faça um algoritmo que receba como entrada as medidas dos dois catetos de um triângulo retângulo e calcule e exiba a medida da hipotenusa e a área do triângulo.

Resumo da aula

- Aprendemos:
  - Utilizar logica com condição
  - Utilizar lógica com repetição
  - Aplicar algoritmo para declarar variáveis
  - Aplicar algoritmo para declarar constantes
- Próxima aula vamos entender mais de lógica para aplicarmos no código;

Obrigado!!!

Profº Jânio Eduardo

janio.vasconcellos@gmail.com